REMARKS

Claims 1-20 were pending at the time of examination. Claims 1, 3-4 and 10-11 have been amended. Claim 2 has been canceled. No new matter has been added. The applicants respectfully request reconsideration based on the foregoing amendments and these remarks.

Objections to the Claims

Claims 11 and 16 were objected to because claim 11 recites "...at least one instant" instead of "... at least one instance," and because claim 16 was incorrectly marked as "Currently Amended." The applicants have changed the of a misspelling of the term "instant" in claim 11 and changed the status of claim 16 to "Original," and submit that the objections to claims 11 and 16 be removed.

Claim Rejections – 35 U.S.C. § 101

Claims 1, 9, and 10-13 were rejected under 35 U.S.C. § 101 as being directed to non-statutory subject matter. Claim 1 has been amended to include the subject matter originally recited in claim 2, which has now been canceled. The applicants submit that claim 1, as amended, now complies with the "practical application test" as described by the Examiner in the Office Action, and that the rejection under 35 U.S.C. § 101 for claims 1 and 9 be removed.

Claim 10 has been amended to recite that the compiler system is embodied in a computer readable medium. The applicants submit that this amendment renders the claim directed to statutory subject matter, and that the rejection under 35 U.S.C. § 101 of claims 10 and dependent claims 11-13 be removed.

Claim Rejections – 35 U.S.C. § 103

Claims 1-7 and 9-20 were rejected under 35 U.S.C § 103(a) as being unpatentable over U.S. Patent No. 6,308,320 to Burch (hereinafter "Burch") in view of U.S. Patent No. 5,408,665 to Fitzgerald (hereinafter "Fitzgerald"). Claim 8 was rejected under 35 U.S.C. § 103(a) as being unpatentable over Buch and Fitzgerald, as applied to claim 1, and in view of U.S. Patent No. 6,041,180 to Perks et al. (herinafter "Perks"). The applicant respectfully traverses these rejections.

Claim 1, as amended recites the steps of:

" identifying one or more **instances** available for use in the **one or more library object files, using linker symbol names for the one or more instances;**

receiving a first request to create **a first instance** during compilation of the source program;

determining whether the **first instance** has been identified in the one or more library object files; and

creating the **first instance** when the determining determines that the first instance has not been identified in the one or more library object files and not creating the first instance when the first instance has been identified in the one or more library object files."

The Examiner alleges that these steps are shown by the "reuse depository" in Burch and the corresponding sections of the Burch specification. The applicants respectfully disagree for at least the following reasons.

The first step of claim 1 requires the identification of one or more **instances**. As can be seen in the background section of the applicants' specification, in particular on page 2, lines 5-20, modern programming languages often provide a mechanism for a programmer to write portions of the source code, which can be replicated and specialized for a number of different uses. This specialization is generated by the compiler in a process referred to as "instantiation," and the specialized code is referred to as an **instance**. Because only one instance is typically required to produce the executable program, a programming system should only produce one "effective" instance for any one program. Thus, in the method for compilation of a source program using one or more associated library object files, as recited in claim 1, the first step identifies one or more **instances** available for use. When the source program is compiled, as recited in the second step, a request to create an instance is received. In the third step, the method determines whether this instance has been identified in the first step of claim 1. Finally, if the instance has not been identified, the instance is created in the fourth step of claim 1, and if the instance has been identified, the instance is not created.

Furthermore, as can be seen in the first step of claim 1, the instances are located in one or more library object files. That is, the instances are contained within the one or more object files. This is further explained on page 9, lines 7-10, of the applicants' specification, which recites that "In one embodiment, the instance name extractor 202 accesses the object file 214, to extract instance names that may be available and stores these instance names in the instance name storage 206." As can be seen in the first step of claim 1, this access is accomplished by **using linker symbol names** for the one or more instances. As a result, in the applicant's invention, the library object files are directly accessible by the compiler, without requiring a significant amount

of preparatory work, e.g., preparation of options files (*see* specification, page 8, lines 8-10). Another benefit resulting from the use of linker symbol names is that it is no longer necessary to transform between linker symbol names and programming language symbol names in order to determine whether an instance already exists in a library object file (*see* specification page 8, lines 13-16).

Turning now to Burch, Burch discloses a "reuse depository," which is a directory that contains a collection of compiled object files that will be reused (see Burch, col. 10, lines 32-34). Nowhere in Burch is it mentioned that the object files in this collection can contain any **instances** available for use, as required by claim 1. Burch also never mentions any of the specific steps that are recited in claim 1, that is, "identifying one or more **instances** available for use in the **one or more library object files**...; receiving a first request to **create a first instance** during compilation of the source program; determining whether the **first instance** has been identified...; and **creating** the **first instance**..."

Furthermore, Burch suffers from many of the drawbacks that are avoided with the applicants' invention. For example, the object files in Burch's repository are generated from "intermediate files 122," which are in turn generated from "source code 118" by "source code compilers 107" (Burch, FIG. 3A). Also, as can be seen in col. 10, lines 46-58, Burch uses a very complex and different method for identifying object files, compared to the method of claim 1:

> "In order to identify object files 120 corresponding to a given intermediate file 122, the intermediate code stream 202 is transformed into a secure hash value 205 that is recorded in both the intermediate code file 122 and the corresponding object code files 120. The secure hash value 206 is derived from the intermediate code stream 202 that reflects the compilation options and computer directives. Additionally, override instructions... may be used to derive the secure hash value 206. The intermediate code stream 202 plus the override instructions 210 uniquely determine a particular object code file 120 generated from the intermediate code file 122."

It ought to be clear that this method of identifying **object files** in the depository of Burch is very different from the claimed method steps of identifying **instances** using **linker symbol names**. It also ought to be clear that even if it were possible to identify instances in Burch's method, no time savings of the type that is accomplished in the applicants' invention would be possible, due to all the extra operations that need to occur, such as generating intermediate files, hash values, compilation options, computer directives, and so on.

The Examiner has cited Fitzgerald in order to cure the deficiency that "Burch does not specify that the depository to create a first instance of object file is library of object files" (Office Action, page 4). The applicants have some difficulty in understanding exactly what the Examiner is referring to in this statement, but regardless it is clear that Fitzgerald does not cure the deficiencies of Burch listed above. Furthermore, there can be no reasonable expectation of success when combining Burch and Fitzgerald, which is required to be shown by the Examiner in order to establish a *prima facie* case of obviousness, since neither of the documents discuss the notion of identifying and using instances. For at least these reasons, the rejection of claim 1 is unsupported by the art and should be withdrawn.

Claims 3-9 are all dependent from claim 1, and the rejection of these claims is unsupported by the cited art for at least the reasons discussed above with regards to claim 1, and should be withdrawn.

In rejecting claim 8, the Examiner combines Burch and Fitzgerald with Perks to establish a *prima facie* case of obviousness. However, Perks does not add anything that cures the deficiencies discussed above, and it is respectfully submitted that the rejection of claim 8 in particular be withdrawn.

Claim 10, as amended, recites "a library object file **including at least one instance available for use by the source program**, the at least one instance being **identifiable by a linker symbol name**;" and "an enhanced compiler suitable for compilation of source code, wherein the enhanced compiler **accesses the library object file** to **identify the one instance available** in the library object file." Both of these limitations recite instances and linker symbol names, which are neither shown in Burch, as discussed above with regards to claim 1. As essentially the same rationale was used by the Examiner in rejecting claim 10 as in rejecting claim 1, the discussion above with regards to claim 1 applies also to claim 10, and the rejection should be removed for at least the same reasons.

Claims 11-13 are all dependent directly or indirectly from claim 10, and the rejection of these claims is unsupported by the cited art for at least the reasons discussed above with regards to claim 10, and should be withdrawn.

Claim 14 recites "instances" and "linker symbol names" in several places, for example, the preamble states that claim 14 is "a method of compilation of source program using one or more associated library object files with instances that are available for use by the source program." Claim 14 further recites the steps of:

"extracting from the linker name table one or more **linker symbol names** that are **likely to correspond to instances**;

storing the one or more linker symbol names that have been extracted as one or more stored linker symbol names;

receiving a first **request to create a first instance** during compilation of the source program, said **first instance having a first linker symbol name**;

comparing the first linker symbol name with the one or more stored linker symbol names; and

creating the first instance only when said comparing indicates that the first linker symbol name is not one of the stored linker symbol names."

The Examiner rejected claim 14 using essentially the same rationale that was used in rejecting claim 1. Thus for at least these reasons presented above with respect to claim 1, the rejection of claim 14 is unsupported by the art and should be withdrawn.

Claims 15-16 both depend from claim 14, and the rejection of these claims is unsupported by the cited art for at least the reasons discussed above with regards to claim 14, and should be withdrawn.

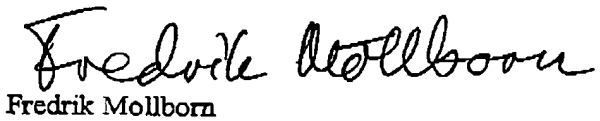
Claim 17 is a *Beauregard* claim corresponding to claim 1. For reasons substantially similar to those set forth above with regards to claim 1, the applicant respectfully contends that the rejection of claim 17 is unsupported by the cited art and should be withdrawn.

Claims 18-20 all depend from claim 17, and are *Beauregard* claims corresponding to claims 3, 4, and 6, respectively. The rejection of these claims is unsupported by the cited art for at least the reasons discussed above with regards to claim 17, and should be withdrawn.

Conclusion

The applicants believe that all pending claims are allowable and respectfully request a Notice of Allowance for this application from the Examiner. Should the Examiner believe that a telephone conference would expedite the prosecution of this application, the undersigned can be reached at the telephone number set out below.

Respectfully submitted,
BEYER WEAVER & THOMAS, LLP



Fredrik Mollborn
Reg. No. 48,587

P.O. Box 70250
Oakland, CA 94612-0250
(510) 663-1100